PRIVACY OF ENCRYPTED VOICE OVER INTERNET PROTOCOL

A Thesis

by

TUNEESH KUMAR LELLA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2008

Major Subject: Computer Science

PRIVACY OF ENCRYPTED VOICE OVER INTERNET PROTOCOL

A Thesis

by

TUNEESH KUMAR LELLA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Riccardo Bettati
Committee Members,	Ricardo Gutierrez-Osuna
	A.L.Narasimha Reddy
Head of Department,	Valerie Taylor

August 2008

Major Subject: Computer Science

ABSTRACT

Privacy of Encrypted Voice Over Internet Protocol. (August 2008)

Tuneesh Kumar Lella, B.E.; M.S., Birla Institute of Technology

Chair of Advisory Committee: Dr.Riccardo Bettati

In this research, we present a investigative study on how timing-based traffic analysis attacks can be used for recovery of the speech from a Voice Over Internet Protocol (VOIP) conversation by taking advantage of the reduction or suppression of the generation of traffic whenever the sender detects a voice inactivity period. We use the simple Bayesian classifier and the complex HMM (Hidden Markov Models) classifier to evaluate the performance of our attack. Then we describe the usage of acoustic features in our attack to improve the performance. We conclude by presenting a number of problems that need in-depth study in order to be effective in carrying out silence detection based attacks on VOIP systems.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

VOIP calls may be considered difficult to intercept, but there are already many tools over the internet that can capture and replay the entire conversations. So VOIP community has come to a general agreement that VOIP communications must be encrypted and in result many scalable and light-weight encryption methodologies have been proposed. [2] and [3] are some among them. To be successful in maintaining the privacy and security of the VOIP conversations, utmost importance should be given to the confidentiality measures such as encryption to make sure that they are robust to both cryptographic and "out-of-the-box" attacks and their capabilities are clearly understood.

In the proposed research we are measuring the ability of the attacker to intercept the encrypted VOIP conversation using the silence suppression feature of VOIP. To compete with traditional phones and to have a wide and scalable deployment, VOIP applications should be able to compete well by conserving bandwidth using silence suppression and other such methods.

It is therefore likely that encrypted VOIP systems will continue to use silence suppression in some way to conserve bandwidth.In this research we will propose traffic analysis methodologies based on Bayesian classification and Hidden Markov models and we will perform a series of experiments to measure how much information can be gathered from an encrypted conversation that uses silence suppression.

The motivation for using traffic analysis is from different sources. The usage of traffic and timing analysis to break confidentiality of the communication sessions

---

The journal model is *IEEE Transactions on Automatic Control*.

has become inevitable these days, as almost all the network communications are involving the use of data encryption and header portions. Wagner *et al.* [4] used the inter-keystroke time intervals of keyboard in SSH connections to infer the typing behavior of users, and are able to crack the SSH passwords entered by the user. They collected the inter-keystroke timings on the keyboard by many users and constructed a Hidden Markov Model using which they are able to crack the passwords. We are applying similar kind of approach in this research. Using the talk spurt lengths of the words, we are using different approaches like simple Bayesian classification and Hidden Markov Models to map the talk spurt lengths back to the spoken words.

CHAPTER II

TRAFFIC ANALYSIS

When the communication between participants in a network application is encrypted and the content of the communication is beyond the reach of crypto-analysis, attackers may want to resort to traffic analysis in order to gain information. Traffic analysis infers information from traffic patterns, which may range from existence or absence of communication to fine-grained timing analysis of packets or messages. The usage of traffic and timing analysis to break confidentiality of the communication sessions has become inevitable these days, as almost all the network communications are involving the use of data encryption and header portions. Traffic analysis is also shown to be applicable against anonymous communication systems [5], [6].

Wagner *et al.* [4] analyzed the inter-packet time intervals in SSH connections to infer the typing behavior of users, and are able to crack the SSH passwords entered by the user. For this they collected keyboard timings and trained a Hidden Markov Model on keyboard inputs from users using which they are able to crack the passwords.

Packet timing analysis has been used in different variations to break anonymous communication systems. In [5] and [6], traffic analysis based attacks on anonymous web servers are described and quantitatively analyzed.The functionality of anonymous web server is such that it acts as a proxy for the web clients and uses encryption and packet header mangling like in a NAT proxy to reduce the correlation between any incoming requests from users and outgoing requests to servers. The fact that number of HTTP objects and their sizes are often revealed is used for this particular attack. Number of objects and object sizes are used as a signature for identifying considerable fraction of the webpages accurately. These signatures can be exploited by the attacker until the web anonymizer takes care of this issue.

The so called MIXes [8] are commonly used in anonymity networks, for example in TOR [7]. These MIXes re-route traffic, encrypt the content and the headers, and perturb the timing and order of packets to make the traceability of connection hard. Attacks exist on these MIXes as well. Serjantov and Sewell [9] analyzed the possibility of a lone flow on an input link of a MIX. If the rate of this lone input flow is approximately equal to the rate of a flow out of the MIX, this pair of input and outflow flows are correlated.

Similarity between sender's outbound traffic and receiver's inbound traffic became a source of analysis for some traffic analysis experts. For example Zhu *et al.* used statistical measures like mutual information to correlate the flows and are successful. Their observation from the TCP flows indicate that too much timing perturbation in the MIXes makes anonymity weak to maintain. Levine *et al.* [11] are also interested in the problem of discovering if two participants are communicating, and they use cross correlation to measure similarity between flows. They proposed a defensive dropping method to thwart the attack. In continuous MIXes, each packet is delayed independently and Danezis [12] uses this fact as a starting point for attack and describes the departure distribution of packets as a convolution of the arrival with delay distribution, which in turn provides the basis for the correlation of incoming and outgoing packets.

Traffic analysis has been used to develop detectors for honeypots or other forms of bots in multiplayer online games. To emulate user behavior (game-bots or crawlers) or system-level latencies (in honeynets) , the bots typically follow a timing pattern. This observation is shown in [17] with the implementation of the timer management mechanism of the operating system which shows the pattern in form of periodicities in inter-response times in honeynets.

Correlation attacks may not be able to scale up when large systems are used. Zhu

and Bettati [13] considered appropriate preconditioning of timing data and they used Blind-Source-Separation [14] to separate the traffic into groups of flows. Manipulation of timing data is also considered by some. For example, Serjantov *et al.* [9] uses "spikes" in the traffic to find the communication relationship between users. Fu *et al.* [15] propose "flow marking" in a wireless setting, where periodic interference patterns are generated, which in turn are visible along the path of the affected connection. A similar mechanism has been proposed by Wang *et al.* [16] to trace VOIP calls. At the sender a slight adjustment is done to the timing of the VOIP packets by delaying packets of different VOIP calls according to different patterns. The authors claim that the disturbance is sufficiently small so as to not be noticed by the end users. At the same time, the timing mark can still be recovered near the receiver so as to identify the originator.

CHAPTER III

MOTIVATION

Traffic Analysis in VOIP is greatly helped by silence suppression. Silence suppression in VOIP aims to save network bandwidth by not transmitting packets when either parties of a VOIP conversation are silent. In order to be widely and scalably deployed, VOIP requires both low bandwidth voice compression and silence suppression. In constraint bandwidth environments, the need for bandwidth conservation is critical. In less constraint environments, the need to be a "good citizen" requires the VOIP to conserve bandwidth as well. For example, Skype relies on relays to route some of its traffic. Since these relays are Skype user nodes, they have to co-habit well with the rest of the applications and traffic at their site. Otherwise, they may attract the attention of system administrators and be shut down. A brief description of silence suppression in some of the widely used VOIPs follows. In the rest of the thesis, we describe how we take advantage of silence suppression to violate VOIP confidentiality.

A. Silence Suppression in VOIP

Each VOIP software uses its own voice codec to code and packetize the voice signals. To conserve the bandwidth used, voice activation detection scheme is used to reduce (e.g. in Skype or GTalk) or suppress (e.g. in SpeakFreely) the sending of packets during silence periods of the conversation. GTalk uses voice activity detection and sends "comfort noise" packets during silence periods and at a slower rate compared to that of during the speech periods.

Skype [18] sends UDP packets at a rate of one every 60 ms during the voice active periods, i.e. when the sender is speaking. Each such packet contains two frames of 30 ms iLBC (Internet Low Bit-rate Codec) [1] compressed speech. During a silence

period, the sender generates UDP packets of 25 bytes each every 100 ms.

Table I. Silence suppression in GTalk

Packet	Delta(ms)	Jitter(ms)	Status
31	0	0	comfort noise (PT=13
33	96.43	40.46	Comfort noise (PT=13
35	104.48	43.84	Comfort noise (PT=13
38	97.99	47.48	Comfort noise (PT=13
41	96.58	50.97	Comfort noise (PT=13
68	61.21	253831.93	[Ok]
72	62.68	297963.02	[Ok]
75	59.88	339336.9	[Ok]
77	58.62	378124.93	[Ok]
83	61.02	414488.12	[Ok]

GoogleTalk [18] also uses voice activity detection and sends "comfort noise" packets during silence periods and at a slower rate compared to that of during the speech periods to save bandwidth. Google Talk uses packetization intervals that are similar to Skype, that is, 60 ms during talk spurts and 100 ms during silence periods. But Google Talk can use a number of codecs and in most of the cases, it uses a variable bit-rate PCM codec that results in speech-dependent packet sizes that are between 120 and 200 bytes. Table I shows the GTalk conversation packet trace as captured by the *Wireshark* [19] network monitor. The trace indicates a talk spurt followed by a silence period. *Wireshark* correctly labels the packets sent during the silence period with a "comfort noise" tag in the plain text trace. Talk spurts can also be distinguished from silence periods by monitoring the inter-packet times (marked "Delta" in the table): At the beginning of the talk spurt, the inter-packet

time drops to approximately 60 ms and when the silence period starts, it returns to approximately 100 ms.

Table II. Silence suppression in SpeakFreely

Packet	Delta(ms)	Jitter(ms)	Status
16	79.99	0.95	[Ok]
17	79.98	0.89	[Ok]
18	79.99	0.84	[Ok]
19	80.00	0.79	[Ok]
21	559.94	30.73	[Ok]
22	79.99	28.31	[Ok]

SpeakFreely [20] is a freeware VOIP that supports a variety of codecs, such as GSM and adaptive differential PCM (ADPCM). GSM, for example, operates at 13 kbit/sec and uses a 80 ms packet period. SpeakFreely uses silence suppression instead of silence detection i.e. it does not send any packets during silence periods, making the silence detection by timing-based traffic analysis very simple. Table II shows the trace of a SpeakFreely voice conversation as captured by wireshark. The data shows a talk spurt (packets 1 to 4) followed by a silence period of 560 ms (packet 5), which in turn is followed by another talk spurt (starting at packet 5).

B. Problem Statement

Although silence suppression can significantly save bandwidth, it can disclose the end users' conversation behavior. In fact, when only the encrypted packet flow is available, talk spurts and other characteristics can be estimated from the packet timings. Due to the encryption of the VOIP flow, we assume that content or size of VOIP packets

is of no relevance. From the packet timestamps, the attacker reconstructs the inter-packet delays and packet rates. In this data, she applies the thresholding to detect voice activity and so is able to construct a series of *estimates* of the talk spurt lengths in the VOIP conversation. The attacker tries to map these talk spurt lengths of words back to the original words spoken.

The objective of this research is to study the effectiveness of the speech recovery by the attacker. This study comprises the description of attack methodology, which involves talk spurt estimation based on packet timings and the speech recovery, and then the evaluation of the effectiveness of the attack followed by the usage of acoustic measures for performance improvement.

Throughout this thesis, we assume that the attacker uses an automated system for data collection and for classification. We assume that a human operator processes the results provided by the classification algorithms (for example by listening to the recovered conversations) and intelligently filters and edits the classification results. In this way, we expect that less than perfect classifiers can still be effective as long as they allow the psycho-receptive mechanism of the human operator to complete the recovery of the conversation.

CHAPTER IV

SYSTEM MODEL AND IMPLEMENTATION

A. System Model

We consider a VOIP communication between a single sender and a receiver. The communication is modeled as a sequence of symbols $s_0, s_1, \ldots, s_i, \ldots$ (each symbol is a spoken word in our case) that are sent over the VOIP connection. The sequence of symbols is presented as input to the VOIP system in form of a continuous voice signal. Let l_i be the length of the voice signal of symbol s_i (the accurate talk spurt length). The accurate talk spurt lengths l_i of the symbols s_i are calculated by detecting the start and the end of the voice spurt in the voice signal using the audio editor software, *audacity* [21]. In the VOIP conversation, the packets are encrypted and likely padded to a fixed size. Hence, in this thesis, we assume that packet sizes are of no use to the observer.

The attacker is assumed to have access to the VOIP conversation link. She "listens" to the VOIP conversation by establishing a wiretap and collecting the timestamps of the packets. Inter-packet delays are calculated by the attacker from the packet timestamps, and a simple thresholding is applied on this data to detect voice activity, and the attacker will be able to construct a series of estimates $l'_0, l'_1, \ldots, l'_i$ of the original spurt lengths $l_0, l_1, \ldots, l_i$ of the symbols. Now that the attacker has collected the estimated spurt lengths, her objective is to reconstruct the original sequence of symbols.

B. Methodology

A set of voice signals, each representing a symbol, is selected from a plain text data by the attacker for training. An important assumption is that the sentences are spoken slowly during training, i.e. there is a minimal silence period between each word in a sentence. For each talk spurt in the sequence, she computes, over all the symbols, the probability distribution $P(s_i|l'_j)$ of a talk spurt being classified as a particular symbol. For each symbol, the attacker is assumed to have access to a sufficient number of occurrences of each symbol. As a result, each symbol s_i has a sufficiently large number of samples $s_i^1, \ldots, s_i^k, \ldots, s_i^n$. The talk spurt length distribution of each symbol thus approximates to a Gaussian distribution. Once the spurt lengths are measured, and the gaussian distributions are estimated, the attacker performs two types of training, which we call *context unaware* and *context aware*.

1. Context-Unaware Training

By "Context unaware" we mean that each symbol in the sequence of words is considered individually and the grammar of the sentence is not considered. The talk spurt lengths $l_i^{(k)}$ of sample $s_i^{(k)}$, both of the direct voice signal and of the voice signal that is sent over the VOIP conversation link, are determined, and the likelihood probability distribution $P(l_i = l|s_i)$ for each talk spurt length l over all the symbols s_i is calculated. A classifier based on Bayesian analysis [22] is used by the attacker to reconstruct the symbol distributions from the collected estimated spurt lengths.

In general, Bayes' theorem [22] is used to compute the probability of an event, given the observation. Given the observation B, the probability of event A is calculated as

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{4.1}$$

Here $P(B|A)$ is called the *likelihood probability* and $P(A|B)$ is called the *posterior probability.*

Using the Bayes' theorem , we calculate the posterior probabilities for each symbol s_i, given the talk spurt length l_i. We classify the observed talk spurt length l as the symbol s^* having the largest posterior value $P(s_i|l)$, where

$$s^* = argmax_{S_i} P(s_i|l) = \frac{P(l|s_i)P(s_i)}{P(l)} \tag{4.2}$$

The classification based on Equation (4.2) is context unaware because $P(s_i|l)$ depends on the likelihood distribution for s_i only.

2. Context-Aware Training

The encrypted transmission of symbols over a VOIP channel can be represented as a *discrete-time Markov-Modulated Process*, where each state (representing the transmission of a symbol) triggers the (observable) emission of a spurt of length l. The *a priori* probability of sequences of symbols to appear in a text can be captured in form of *transition probabilities* between states in the Markov model. Since only the sequence of talk spurt lengths $l_1, l_2, \ldots$ is observable, while the sequence of *transmitted* symbols remains hidden, we call such a Markov model a *Hidden Markov Model* (HMM).

More formally for our problem, the HMM is formulated as follows:

- Let $\mathcal{S}$ be a set of states that emit output symbols in $\mathbb{R}$.
- Let $\mathcal{A} = a_{k,i}$ be a $|\mathcal{S}| \times |\mathcal{S}|$ matrix of state transition probabilities.
- Finally, let $\mathcal{O} = \{O_k\}$ be a set of continuous random variables, one for each

state in $\mathcal{S}$.

Sending encrypted symbols over a VOIP channel then corresponds to the following HMM:

- $\mathcal{S} = \{S_1, S_2, \ldots, S_N\}$, corresponding to the set of symbols.
- $\mathcal{A}$, corresponding to the *a priori* sequences of words in the underlying language of communication over the VOIP channel.
- $\mathcal{O} = \{O_1, O_2, \ldots, O_N\}$, where O_i corresponds to the length distribution of talk spurts for symbol S_i.

A *path* $\pi = \pi_1 \cdots \pi_n$ in the HMM is a sequence of states and represents a sentence transmitted over the VOIP channel. The attacker observes the sequence $o = o_1 \cdots o_n$. Given that the sequence π is *hidden* from the attacker, the latter faces a *Decoding Problem*, that is, she has to find an optimal path $\pi^* = argmax_\pi P(o|\pi)$ for o such that $P(o|\pi)$ is maximized.

In this training step the transition matrix $\mathcal{A}$ is constructed based on the sequences of symbols (words) in the training sentences, and the random variables in $\mathcal{O}$ are borrowed from the previous, context-unaware training step.

During the attack, the observer uses dynamic programming, for example the Viterbi algorithm [24], to estimate an optimal path based on the observed spurt lengths. In this way she reconstructs the sequence of words transmitted over the VOIP channel.

In summary, the attacker does not rely on packet content, or packet sizes in her attempt to reconstruct the sequence of words transmitted. Instead, she relies on packet timing information only, in order to identify talk spurt boundaries.

CHAPTER V

EXPERIMENTAL RESULTS

To evaluate the effectiveness of the attack described in the previous chapter, we conducted a sequence of experiments both for the (admittedly unrealistic) case where accurate talk spurt lengths (measured length of voice signals in units of time) are available and for the case where the talk spurt lengths have been estimated from analysis of packet timings. The ability to map the talk spurt lengths back to the original symbols (words) is considered as the measure of performance. The performance measures are calculated for both the cases- Context-unaware case that uses a Bayesian classifier to estimate the symbols and Context-aware case that uses a HMM based classifier on training sentences.

We used the following rhymes for our experiments. For simplicity, all the words are treated separately i.e. there is a silence period after each word in the sentence.

> Rhyme 1: "Betty bought some batter butter. But she found the butter bitter. So she bought some better butter to make the bitter batter better."
>
> Rhyme 2: "Popeye the sailor man, lived in a garbage can, Ate all the worms and spat out the germs; the best a man can."
>
> Rhyme 3: "My older brother is a pest, He loves to sit in the west, He gets no rest, but pins his chest, to the lap top vest."

New sentences are formed from the words in these rhymes and the corresponding sequence of talk spurt lengths are fed to the classifiers, which will emit the output sequence of symbols. Rhyme 1 allows us to separately evaluate the effectiveness of context free symbol classification (e.g. the ability to separate "but" from "butter" purely based on the spurt length) and the improvement we get after adding context

awareness (e.g. the ability to separate similar sounding words like “better”or “bitter” from “butter” or “batter”, which cannot be separated based on talk spurt length only). Rhyme 2 and Rhyme 3 combined as a single database of sentences are used to verify the correctness of results obtained with the Rhyme 1 i.e. whether we get similar results with a different set of sentences.

Each rhyme is repeatedly spoken, 25 times to calculate the gaussian distributions of the talk spurt lengths for each symbol (word) in the database. The transition probabilities required for the context-aware classifier are calculated from a set of newly formed sentences (a total of 16 such sentences) from the words of Rhyme 1 above. Examples of such formed sentences are “Betty bought the bitter butter”, “She bought the bitter butter”, “Make the bitter butter better” etc. Similarly, 25 new sentences are generated from the words of Rhyme 2 and Rhyme 3.

A. Talk Spurt Analysis with Accurate Talk Spurt Lengths

As a base line we assume that the attacker has access to the accurate lengths of talk spurts. For this, during the context-unaware portion of the training we use accurate spurt lengths that have been determined by detecting the start and the end of the voice spurt in the (non-packetized) voice signal.

Figure 1 shows the spurt length distribution of 50 samples each of the words “bought”, “some”, and “batter”. From this figure we can clearly see that some symbols (for example “bought” and “batter”) can be clearly separated based on the spurt length, while others (for example “bought” and “some”) have similar spurt lengths and their talk spurt length distributions overlap.

We calculated confusion matrices using talk spurt length distributions from 25 samples for each symbol for random guess classifier, Bayesian classifier and HMM

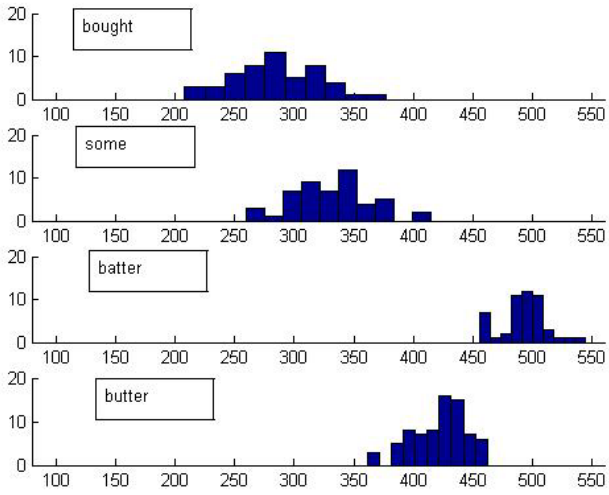

Fig. 1. Accurate spurt lengths of voice signals

based classifier. In the confusion matrix, an entry, for example $c_{i,j}$ signifies the probability that Symbol S_i is classified as Symbol S_j. If the classifier is perfect, then C would be an identity matrix. For this, we randomly feed one of the 16 training sentences to the classifier and calculate the corresponding confusion matrix based on the input and the output sequences. For the HMM classifier, we use the remaining 15 sentences for training. All the 16 sentences are either 4 words or 5 words long and there are totally 14 words in the Rhyme 1. For example, to get an classification error of 0.2, the random guessing should guess 4 words wrongly in a sentence of 5 words long. The corresponding probability is $C_{5,4} * (13/14)^4 * (1/14)$. The performance of random classifier suffers badly as the total number of words increase. We obtained a corresponding confusion coefficient by adding all the elements $c_{i,j}$ except the diagonal ones ($c_{i,i}$), for each of the confusion matrices from the 3 classifiers. The less the confusion coefficient, the better the performance of the classifier. Figure 2 shows the comparison of the confusion coefficients for the random, Bayesian and HMM based classifiers. Confusion coefficient is less for the HMM classifier which makes the HMM

classifier better than the Bayesian and random classifiers.

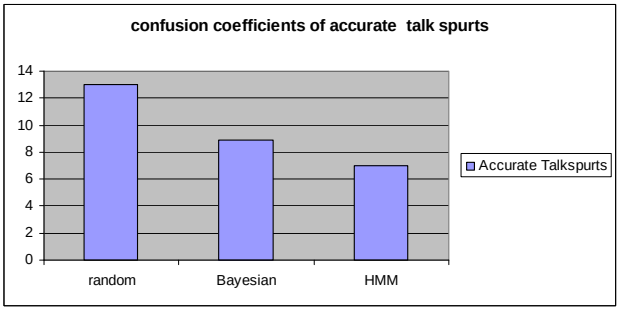

Fig. 2. Comparing confusion coefficients of random, Bayesian and HMM(15) classifiers on sentences using accurate spurt lengths of Rhyme 1

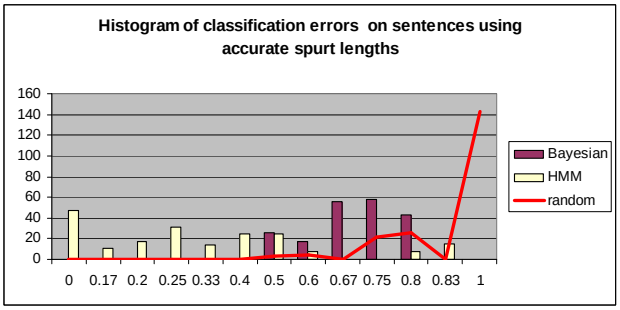

Fig. 3. Histogram of classification errors using random, Bayesian and HMM(15) classifiers on sentences using accurate spurt lengths of Rhyme 1

Figure 3 compares the effectiveness of the Bayesian Classifier against the HMM classifier with 15 training sentences in correctly recognizing a spoken sentence. For this, we randomly feed one of the 16 training sentences to the classifier and record the percentage of erroneously classified words. Performance of random guessing, which is obviously poor, is also included for comparison. The histograms clearly illustrate the benefits of context-aware training using HMM. HMM classifier worked far better than the Bayesian classifier and there are also instances in which all the words are

detected correctly in a sentence (zero error rate case).

Accurate talk spurt length distributions for words from Rhyme 2 and Rhyme 3, and classifiers' performance comparison results obtained with the sentences formed from the combination of these words, can be found in appendix A. We got similar results for this case as well. The Matlab code used for HMM and Bayesian classification can be found in Appendix B.

B. Talk Spurt Analysis on Packetized Voice Signal

When only the encrypted packet flow is available, the talk spurt lengths must be estimated from the packet timing. In these experiments we focused on GTalk. First we establish a VOIP connection between two GTalk users. Then the sender starts speaking the preselected sentences with gaps between words over the VOIP connection. She repeats the sentences several times to have a Gaussian distribution for the estimated talk spurt lengths of each symbol, which is essential for the classification of talk spurt lengths as symbols. For simplicity, we use *Wireshark* [19] at the receiver to capture packets from the sender, and we use the timestamp information to identify talk spurts. Figure 4 illustrates the procedure in which we estimate the talk spurt lengths.

A description of trace of GTalk conversation is given in Chapter III, with Table I as reference. Delta column shows the relative timings when the packets are sent. We use the values in the interval 95-105ms for the *delta* column as a threshold to mark the start time of silence and finish time of talk spurt. Similarly, we use the values in the interval 55-65ms as a threshold to mark the start time of the talk spurt and finish time of silence. Then we calculate the talk spurt lengths by finding the difference between the corresponding sum of timings (in the *delta* column) until the

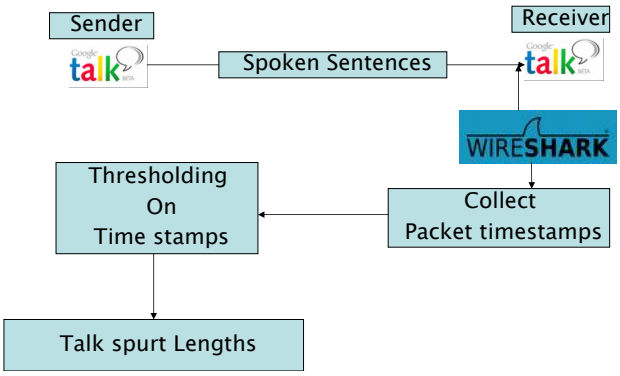

Fig. 4. Talk spurt length estimation procedure

talk spurt finish time and the sum of timings until the talk spurt start time.

We repeat the same series of experiments as for the case of accurate spurt length measurement i.e. measuring the effectiveness of classifiers in detecting the sentence correctly.

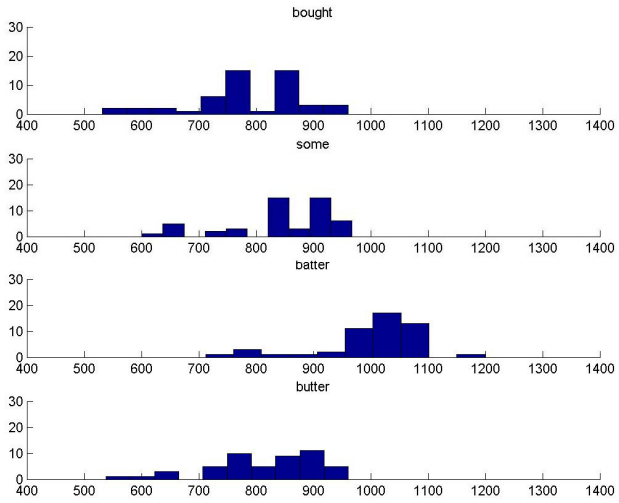

Fig. 5. Estimated spurt lengths based on packet timing in GTalk

Figure 5 illustrates how packet-timing based voice activity detection is a very

noisy estimator for spurt lengths, to a level where simple Bayesian classifier becomes difficult to apply. We see how symbols with some overlap in their talk spurt distributions at voice signal level (for example "bought" and "some") become nearly impossible to distinguish at packet-timing level. Symbols that have no overlap at voice-signal level (for example "bought" and "batter") now become more difficult to distinguish at packet-timing level.

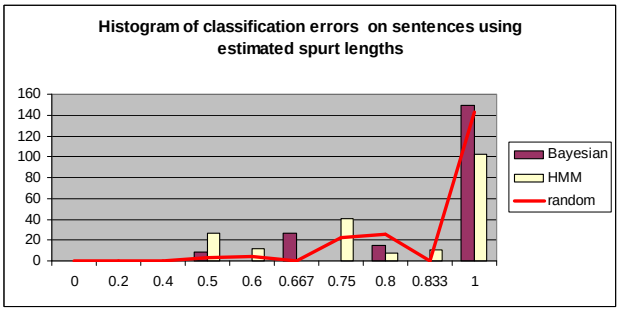

Fig. 6. Histogram of classification errors using random, Bayesian and HMM(15) classifiers on sentences using estimated spurt lengths of Rhyme 1

Figure 6 illustrates the poor performance of the classification with estimated spurt length data. In this experiment, the HMM based classifier performs somewhat better than the ones based on Bayesian classifier and random guessing.

Estimated talk spurt length distributions for words from Rhyme 2 and Rhyme 3, and classifiers' performance comparison results obtained with the sentences formed from the combination of these words can be found in Appendix A.

Two factors that are inherent to the silence detectors applied in the VOIP system contribute to the poor performance of spurt length estimation: First, a *silence threshold* level is used in the silence detector to determine voice activity. If the strength of a speech signal is below the threshold, the speaker is regarded as being silent. Obviously if the threshold is too low or too high, the accuracy of the talk spurt estimator

suffers. This effect is illustrated by the outliers in the histograms in Figure 5, which were caused by extraneous noise, i.e. the acoustic noise around the microphone, in the experiment. Second, the *hang-over time* of the silence detector in the sender negatively affects the accuracy of our estimator. Hang-over time is the duration by which a silence detector delays its final decision to terminate the talk spurt, and its main purpose is to avoid end clipping of speech. Similar to the silence threshold, hang-over time affects both accuracy of our estimator and bandwidth saving. These effects are particularly strong in these experiments because of the short spurt lengths (i.e. multiples of few packet intervals). We expect that these effects are not as strong in longer bursts. In such cases, however, the set of symbols will be very large, thus rendering the classification more difficult.

C. Improvement with n-Viterbi

The talk spurt length distributions for many word pairs are naturally overlapping, and this is particularly the case when the lengths are estimated from the packet timings, which can be clearly seen in Figure 5. This affects the performance of the Bayesian classifier and HMM based classifier as well.

Fortunately in our case, a human operator operates the classification results and filters out erroneous classifications. Classification errors are frequently easily detectable by the human operator, since the resulting sentence makes little or no sense. Therefore it is sufficient for a classifier to generate a small set of "candidate" classifications, from which the presumably intelligent operator picks the most appropriate. In these experiments we make use of the n-Viterbi algorithm, an extension to the 1-Viterbi that simultaneously generates a selection of solutions rather than a single one. For a given value for n, n-Viterbi returns n solutions, from which the human

operator picks the most appropriate one. The value of n is 4 in our case. From the 4 output sequences, the sequence with the maximum number of correctly classified words is selected for measuring the performance.

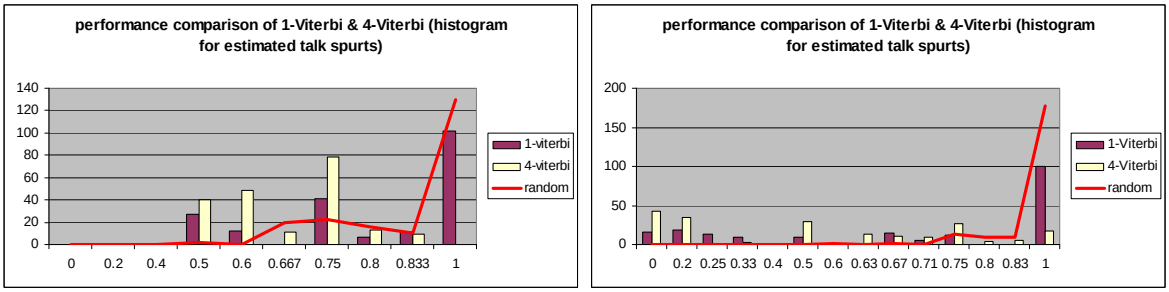

Fig. 7. Comparing performance of 1-Viterbi and 4-Viterbi on sentences using estimated spurt lengths from Rhyme 1 and combination of Rhyme 2 and Rhyme 3

Figure 7 shows the improvement in the performance with n-Viterbi compared to that of 1-Viterbi. The histogram on the left shows the performance from the sentences using Rhyme 1 and the histogram on the right shows that of the sentences from the combination of Rhyme 2 and Rhyme 3. There are no instances in which all the words are classified incorrectly with 4-Viterbi in the case of Rhyme 1 and the number drastically reduced in the second case, which is a major improvement over 1-Viterbi. The significant improvement in the second case can be attributed to the condition that words are less similar to each other compared to that of first case and hence resulting in talk spurt length distributions less overlapping.

CHAPTER VI

IMPROVEMENT OF RESULTS USING ACOUSTIC FEATURES

A. Acoustic Features

The experiments in Chapter V measure the performance of the classifiers in terms of their effectiveness to correctly classify symbols. The error rates in Figure 6 and Figure 7, for example, are calculated as follows:

- Compare the test sentence and the output from the classifier word by word
- For each word, if the classifier detects the word correctly, then the error is zero, otherwise, the error is one.

While this measure is appropriate for speech recognition applications where the symbols must be correctly identified by an automated system, it is much less so for our setting, where a human operator observes and filters the results generated by the classifier. In our case, the operator naturally ignores minor errors in the classification, for example errors between acoustically similar words such as "butter" for "batter". As a result, the problem at hand is less a *speech recognition* problem rather than an *audio query* problem: The classifier returns a list of likely sentences that correspond to the input spurt lengths. If the result is returned in acoustic form, the operator will be able to identify the original sentence easily if it is offered, and can ignore minor errors in the classification.

Simply counting classification errors in such a setting does not provide a measure for the "usefulness" of the classifiers to the operator. More appropriately, any effectiveness measure must penalize errors according to their effect on the ability of the operator to reconstruct the original sequence of symbols.

In the following, we will be using acoustic similarity as a heuristic measure for the ability of the operator to naturally correct classification errors. The rationale is that the operator can more easily correct errors when the symbols sound similar (such as "butter" instead of "better"), than when the symbols sound distinctly different (such as "better" and "the"). Similarity measure is the indication of closeness of the audio signals being compared. We considered the dominant feature vectors based similarity measure [25] used for calculating similarity measure between words and Bhattacharyya distance [26] used for calculating distances between classes. A brief explanation of both the similarity measures is given in the following two subsections.

1. Dominant Feature Vector Based Similarity Measure

We first extract the features from the words and then the similarity measure is calculated between the words using the dominant features. Audio features that are considered are 13 Mel Frequency Cepstral Coefficients (MFCCs) [27], short time energy [28] and zero cross rate [28]. MFCCs represent the spectral envelope of an audio signal. The zero-crossing rate is the rate at which the signal changes from positive to negative or vice-versa. Short-Time energy is a simple short-time speech measurement that can in a way distinguish between voiced (voice active period) and unvoiced (silence period) speech segments, since unvoiced speech has significantly smaller short-time energy. Each audio clip is divided into N audio frames. An n-dimensional feature vector is extracted from each frame and normalized to zero mean and unit variance. Thus, the audio clip is represented as a $n \times N$ matrix [25]. Dominant features are obtained by computing the Eigen decomposition on the covariance of frame based feature vector.The reason for doing Eigen-decomposition is to find the vectors that describe the characteristics of audio clip in the best possible manner. Eigen vectors associated with large Eigen values represent most of the information, and hence they

can be considered as dominant feature vectors.

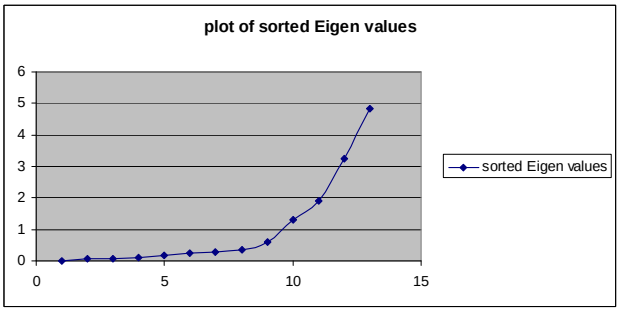

Fig. 8. Plot of sorted Eigen values

Figure 8 shows the plot of sorted Eigen values for a sample audio clip. From the figure, we can be sure that the top 8 Eigen vectors contribute most to the audio clip. The number of Eigen vectors over the entire set of words in our experiments are selected by observing the Eigen values and the value is selected as 8. The equation below is used to calculate the similarity measure between audio clips (words in our case).

$$S = \sum_{i=1}^{n1} \sum_{j=1}^{n2} w_{i,j} s_{i,j}, \quad \text{where} \tag{6.1}$$

$$w_{i,j} = \frac{p_i}{\sqrt{\sum_{i=1}^{n1} p_i^2}} \frac{q_i}{\sqrt{\sum_{i=1}^{n2} q_i^2}}, \quad \text{and} \tag{6.2}$$

$$s_{i,j} = \frac{||X_i^T Y_j||^2}{||X_i||^2 ||Y_j||^2} \quad . \tag{6.3}$$

The values p_i and q_j are the Eigen values of the i^{th} and j^{th}. X_i and Y_j are the corresponding Eigen vectors. The similarity measure is calculated by giving more weight to the Eigen vectors with large Eigen values. The weighted sum is designed such that if the two audio clips are the same, then the weighted sum should be equal

to one. The similarity measure value will be between zero and one. The higher the similarity between the audio clips, the closer the value will be to one. The Matlab code for the computation of similarity measure can be found in Appendix B.

The phonetics involved in any word can be known easily using a dictionary. We can use Acoustic-Phonetic Continuous Speech Corpus like *TIMIT* [29] to compare phonemes of words and compute the distance between the words. This can serve as an alternative to similarity measure.

2. Bhattacharyya Distance

The Bhattacharyya distance [26] is a theoretical distance measure between two Gaussian distributions. It is computationally simple and can be used to derive an upper bound on the optimal Bayesian classification error probability between two classes. The Bhattacharyya distance D_{bhat} is defined as follows

$$D_{bhat} = \frac{1}{8}(M_2 - M_1)^T[\frac{\sum_1 + \sum_2}{2}]^{-1}(M_2 - M_1) + \frac{1}{2}\ln\frac{|\frac{\sum_1 + \sum_2}{2}|}{\sqrt{|\sum_1||\sum_2|}} \ . \tag{6.4}$$

M_is and $\sum_i$s are the means and covariance matrices of the two Gaussian distributions. Optimal Bayesian classification error between two classes with equal a priori probability is bounded by the expression $\varepsilon_{bhat} = 0.5 * \exp(-D_{bhat})$.

The optimal Bayesian classification error can be used to gauge how inherently difficult the classification problem is. We will use these similarity measures to determine the penalties associated with the classification errors.

B. Experiments

1. Dominant Feature Vector Based Similarity Measure

We have done the similar experiments that we have done in the previous chapter. The difference here is the error measurement method. similarity measure $s_{i,j}$ is an indication of closeness of the audio clips and the value of similarity measure always varies between zero and one. Hence $distance_{i,j} = 1 - s_{i,j}$ can be considered as penalty of classification error between the audio clips.

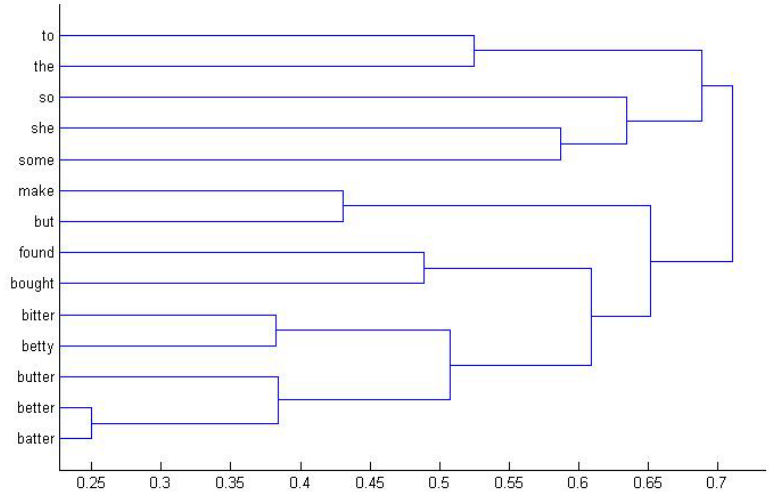

Fig. 9. Dendrogram of $distance_{i,j}$ between symbols of Rhyme 1

Figure 9 shows the dendrogram for the $distance_{i,j}$ between words of Rhyme 1. Clear observation is that similar words like "butter", "better" and "batter" are very near to each other. We considered random, Bayesian, HMM (1-Viterbi), HMM (4-Viterbi) and optimal classifiers to show the improvement in performance. Optimal means the classifier detects all the symbols in the sentence correctly and it is the maximum performance we can obtain using similarity measure based performance evaluation. We aforetime store the acoustic representations of all the utterances of

all the symbols. For each symbol, we elect the centroid sample as the representative signal from the collection of the sample signals. We measure the optimal performance by calculating the average of the similarity measures between the representative voice signals of the output sequence of symbols and the corresponding voice signals of the test sequence. Random classifier classifies the symbols on a random basis out of the words in the Rhyme 1 and the average $distance_{i,j}$ is calculated between the input and the output sequence of symbols. In the similar way, we measure the performance for the Bayesian, 1-Viterbi and 4-Viterbi cases.

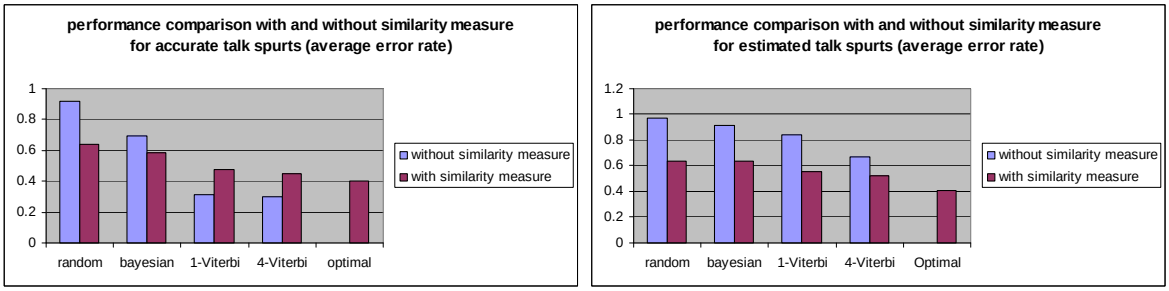

Fig. 10. Similarity measure improvements for symbols of Rhyme 1 with accurate and estimated talk spurts

Figure 10 shows the improvement of the results with similarity measures for performance evaluation for the estimated talk spurts of Rhyme1. Ideally, the optimal performance value should be equal to zero. But it in reality, the similarity measure will not equal zero between different pronunciations of the same word and hence there is a lower threshold the performance can reach. But an intelligent human operator can easily differentiate between different pronunciations of the same word easily and this is not a major issue.

2. Bhattacharyya Distance

From the distributions of symbols (words) that we have already used in the experiments in the previous chapter, we calculated the pairwise Bhattacharyya distances between the symbol (words) distributions and calculated the respective optimal Bayesian classification errors between classes.

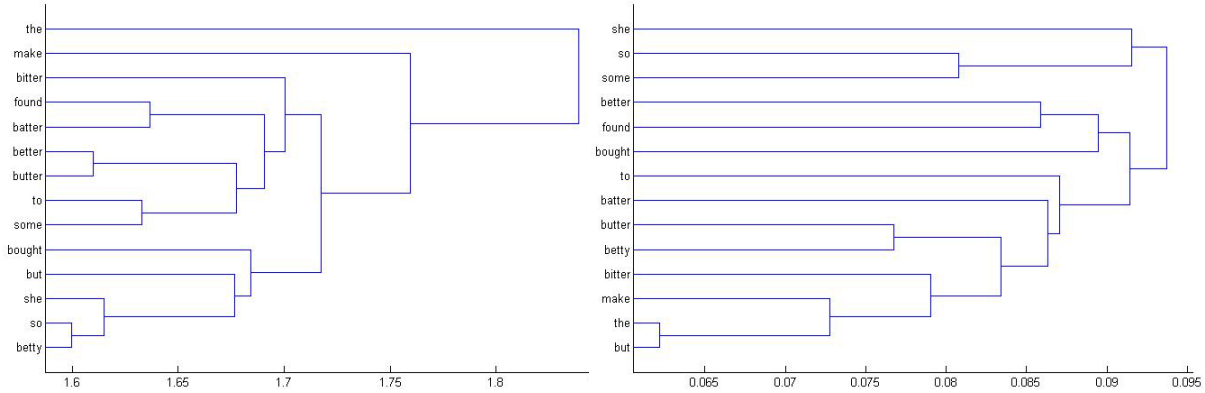

Fig. 11. Dendrograms for Bhattacharyya distances and error rates for symbols of Rhyme 1

Figure 11 shows the dendrograms of the Bhattacharyya distances between symbols and the corresponding classification errors from the Rhyme 1. It can be clearly seen that the similar words like "butter", "better" and "batter" have small pairwise distances and large Bayesian classification errors. Similar is the case with the symbols from Rhyme 2 and Rhyme 3.

We selected three classifiers namely random, Bayesian and HMM classifiers to show the improvement of the results with errors based on Bhattacharyya distance. Random classifier classifies the symbols in the sequence (sentence) randomly, which is the worst case behavior. We calculated confusion matrices for each of the cases and obtained a corresponding confusion coefficient by adding all the elements $c_{i,j}$

except the diagonal ones ($c_{i,i}$). In random classification, $c_{i,j} = 1/\text{n}$, where n is the total number of symbols. Thus, the confusion coefficient in random case would be n(1-(1/n)) = n-1.

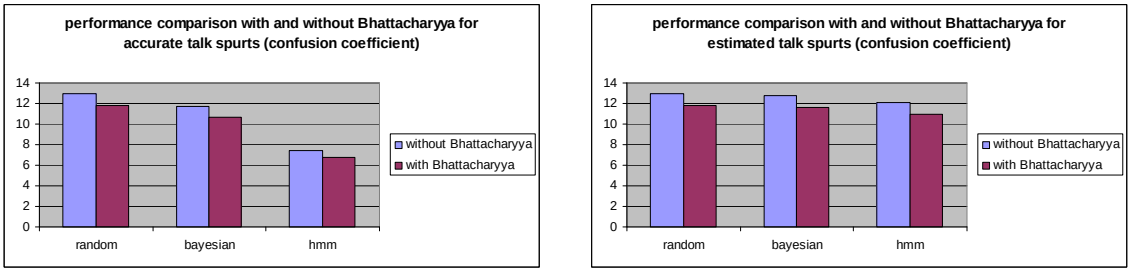

Fig. 12. Bhattacharyya improvements for symbols of Rhyme 1 with accurate and estimated talk spurts

Figure 12 shows the improvement with errors, calculated from Bhattacharyya distance data from symbols of Rhyme 1. In the *without Bhattacharyya* case, we add all the elements except the diagonal ones in the confusion matrix to get the confusion coefficient and in the *with Bhattacharyya* case (using Bhattacharyya data), each element, except the diagonal elements, $c_{i,j}$ is multiplied with the corresponding probability of classifying symbol i as j, $1 - \varepsilon_{bhat_{i,j}}$ and the products are added to get the confusion coefficient.

Then we weighed the errors obtained with the absolute calculation in previous Chapter V with the errors obtained from the Bhattacharyya distances between symbols. Figure 13 shows the phenomena of weighing. There is a clear improvement in the performance of all the classifiers, including random guess classifier, with Bhattacharyya distance based evaluation. HMM based classifier clearly dominated other classifiers.

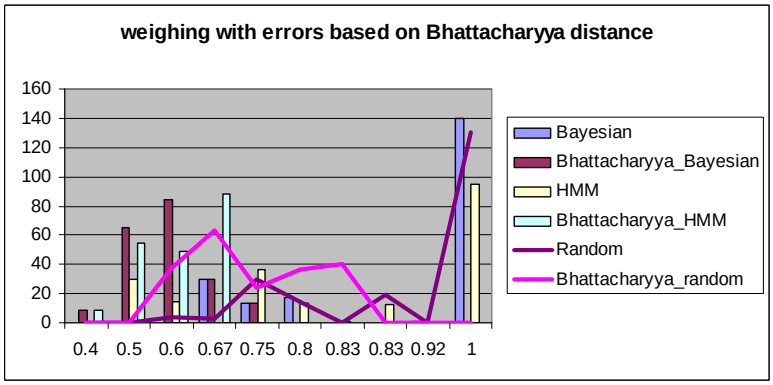

Fig. 13. Weighing the originally obtained errors with Bhattacharyya distance based errors

CHAPTER VII

CONCLUSION AND FUTURE WORK

On one hand, encryption of VOIP traffic is critical for the security and privacy of the service participants. On the other, silence suppression is important for the deployability of VOIP services.

In this thesis, we investigated the effectiveness of the encryption in presence of silence suppression. We presented a model in which attacker uses just the packet timing information of the VOIP conversation to reconstruct the communication content. The first observation is that with accurate talk spurt lengths, the performance of the model is good in constructing the sentences spoken both by using context-unaware Bayesian classifier and context-aware HMM based classifier, HMM obviously dominating the Bayesian classifier. However, with the estimated talk spurt lengths, the performance suffered badly with both Bayesian and HMM based classifiers. The major reason for performance loss can be attributed to too much overlapping of estimated talk spurt lengths of the symbols. In our case, a human operator filters out the erroneous classifications and hence we used n-Viterbi algorithm instead of 1-Viterbi algorithm in HMM and we found a significant amount of improvement in the performance.

The problem we are dealing with is not a speech recognition problem and is more similar to an audio query problem. If the result is returned in acoustic form, the operator will be able to identify the original sentence easily if it is offered, and can ignore minor errors in the classification. Hence, we resorted to acoustic measures for our performance evaluation. We used dominant feature vector based similarity measure between words and Bhattacharyya distance between word distributions as a measure for performance evaluation instead of using the absolute method of one in case of successful detection of word or zero in case of error. The reason is simple. We

are dealing with a similar problem like audio querying and getting similar results is considered a success. For example, if the word “batter” getting detected as “better”, the human operator is assumed to presumably intelligent to correctly identify the word in the output sequence. There is a considerable improvement in the results with the acoustic measures.

We are dealing with sentences with gaps between words, i.e. there is a small silence gap between every word in the sentences spoken. The extension of this work should be to check the effectiveness of the attack with sentences spoken without any gaps between words. Also the use of other information like packet sizes, Jitter etc and whether there will be any impact of these on the estimation of talk spurt lengths need to be investigated.

REFERENCES

[1] S. Anderson, A. Duric, Telio, H. Astrom, R. Hagen *et al.*, "Internet low bit rate codec (iLBC)," Available: http://www.ietf.org/rfc/rfc3951.txt, December 2004.

[2] P. Zimmermann, A. Johnston, and J. Callas, "Zrtp: media path key agreement for secure RTP," Available: http://www.ietf.org/ietf/1id-abstracts.txt, March 2007.

[3] "The Zfone Project," Available: http://zfoneproject.com/, March 2007.

[4] D. X. Song, D. Wagner, and X. Tian, "Timing analysis of keystrokes and timing attacks on SSH," in *Proc. 10th Conference on USENIX Security Symposium (SSYM'01)*, August 2001, pp.25-25.

[5] Q. Sun, D. Simon, Y. Wang, W. Russell, N. Padmanabhan, and L. Qiu, "Statistical identification of encrypted web browsing traffic," in *Proc. IEEE Symposium on Security and Privacy*, 2002, pp.19-30.

[6] A. Hintz, "Fingerprinting websites using traffic analysis," Available: http://guh.nu/projects/ta/safeweb/safeweb.html, 2002

[7] "Tor: Anonymity Online," Available: http://www.torproject.org/, 2004

[8] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," in *Communications of the ACM*, vol. 4, no. 2, pp.84-90, February 1981.

[9] A. Serjantov and P. Sewell, "Passive attack analysis for connection-based anonymity systems," in *Proc. European Symposium on Research in Computer Security (ESORICS)*, 2003, pp.116-131.

[10] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "On flow correlation attacks and countermeasures in mix networks," in *Proc. Workshop on Privacy Enhancing Technologies (PET2004)*, 2004, pp.207-225.

[11] B. N. Levine, K. Reiter, C. Wang, and M. Wright, "Timing attacks in low-latency mix-based systems," in *Proc. Financial Cryptography*, 2004, pp.251-265.

[12] G. Danezis, "The traffic analysis of continuous-time mixes," in *Proc. Privacy Enhancing Technologies workshop (PET 2004)*, May 2004, pp.35-50.

[13] Y. Zhu and R. Bettati, "Unmixing mix traffic," in *5th Workshop on Privacy-Enhancing Technologies*, Cavtat, Croatia, May 2005, pp.110-127.

[14] C. Jutten and J. Herault, "Blind separation of sources, Part 1: An adaptive algorithm based on neuromimetic architecture," in *Signal Process.*, vol. 24, no. 1, pp.1-10, July 1991.

[15] X. Fu, Y. Zhu, B. Graham, R. Bettati, and Zhao, "On flow marking attacks in wireless anonymous communication networks," in *IEEE International Conf. Distributed Computing Systems (ICDCS- 2005)*, Columbus, OH, June 2005, pp.493-503.

[16] X. Wang, S. Chen, and S. Jajodia, "Tracking anonymous peer-to-peer voip calls on the internet," in *Proc. 12th ACM Conference on Computer and communications security (CCS 05)*, New York, 2005, pp.81-91.

[17] Y. Guan, X. Fu, R. Bettati, and W. Zhao, "Netcamo: Camouflaging network traffic for QoS guaranteed mission critical applications," *IEEE Trans. Systems, Man and Cybernetics*, vol. 31, no. 4, pp.253-265, July 2001.

[18] B. Sat and B. W. Wah, "Analysis and Evaluation of the Skype and Google-Talk VOIP Systems," in *IEEE International Conf. Multimedia and Expo, 2006*, Toronto, ON, Canada, July 2006, pp.2153-2156.

[19] "Wireshark frequently asked questions," Available: http://www.wireshark.org/, 2006.

[20] "SpeakFreely for Windows," Available: http://www.speakfreely.org/, February 2002.

[21] "Audacity documentation and support," Available: http://audacity.sourceforge.net/help/, 2007.

[22] "Statement of Bayes' theorem," Available: http://en.wikipedia.org/wiki /Bayes´_theorem/, March 2008.

[23] "Matlab tutorial," Available: http://www.mathworks.com/academia/student_ center/tutorials/launchpad.html, October 2006.

[24] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 267-296, February 1989.

[25] J. Gu, L. Lu, R. Cai, H. J. Zhang, and J. Yang, "Dominant feature vectors based audio similarity measure," in *Fifth IEEE Pacific-Rim Conf. Multimedia (PCM'04)*, Tokyo Waterfront City, Japan, December 2004, pp. 890-897.

[26] B. Mak and E. Barnard, "Phone clustering using the Bhattacharyya distance," in *Fourth International Conf. Spoken Language*, Philadelphia, PA, June 1987, pp.2005-2008.

[27] "PLP and RASTA (and MFCC, and inversion) in Matlab using melfcc.m and invmelfcc.m," Available: http://labrosa.ee.columbia.edu/matlab/rastamat/, July 2006.

[28] M. Greenwood and A. Kinghorn, "Suving: Automatic silence/unvoiced/voiced classification of speech," Department of Computer Science, The University of Sheffield, UK, 1999.

[29] "TIMIT acoustic-phonetic continuous speech corpus," Available: http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1, 2006.

[30] T. Lella and R. Bettati, "Privacy of encrypted voice-Over-IP," in *IEEE International conf. Systems, Man and Cybernetics (SMC 2007)*, Montreal, Canada, October 2007, pp.3063-3068.

APPENDIX A

DATA AND EXPERIMENTAL RESULTS FROM WORDS OF SECOND AND THIRD RHYMES

The spurt length distribution data, and performance comparison of Bayesian and HMM classifiers followed by the improved results using acoustic features of words for accurate talk spurt sengths as well as estimated talk spurt lengths of words from second and third rhymes are presented here.

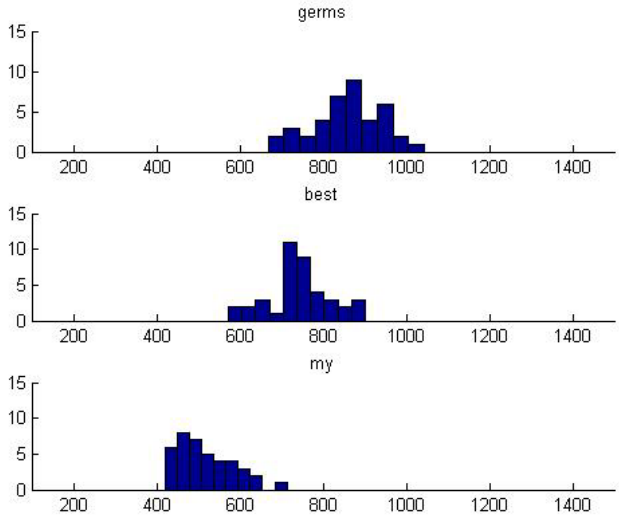

Fig. 14. Accurate spurt lengths of voice signals of symbols from Rhyme 2 and Rhyme 3

In Figure 18, the number 22 signifies the bin which contains all the similar sounding words like 'pest', 'vest', 'chest'. It is obvious as the value of $distance_{i,j}$ is less between similar words.

In Figure 20, the number 22 on the left part signifies the bin which contains all the similar sounding words like 'pest', 'vest', 'chest'. This result is obvious as the value of Bhattacharyya distance is less between similar words.

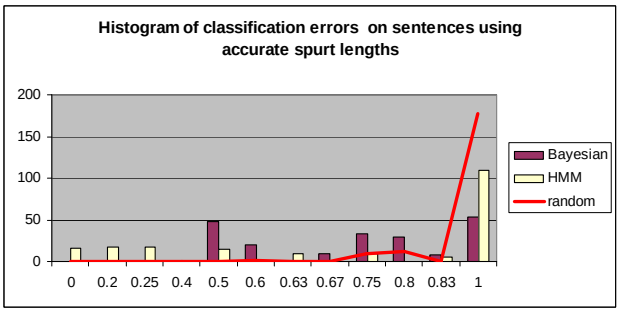

Fig. 15. Comparing Bayesian and HMM(24) classifiers on sentences using accurate spurt lengths of symbols from Rhyme 2 and Rhyme 3

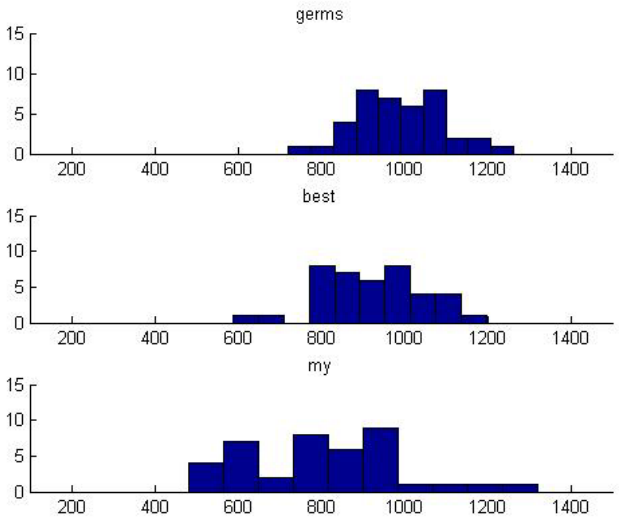

Fig. 16. Estimated spurt lengths of symbols from Rhyme 2 and Rhyme 3 based on packet timing in GTalk

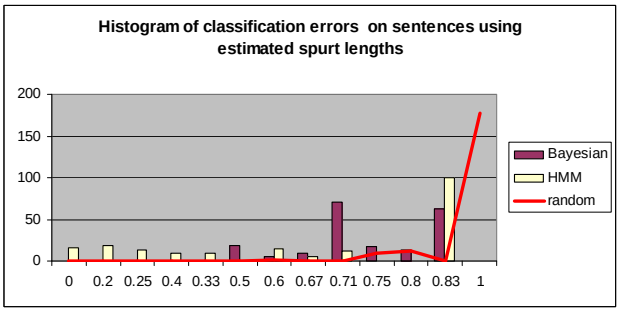

Fig. 17. Comparing Bayesian and HMM(24) classifiers on sentences using estimated spurt lengths of symbols from Rhyme 2 and Rhyme 3

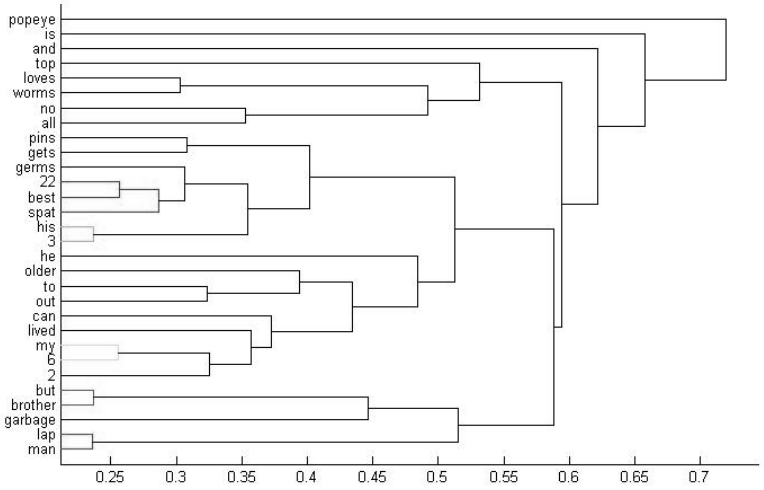

Fig. 18. Dendrogram of $distance_{i,j}$ between symbols of Rhyme 2 and Rhyme 3

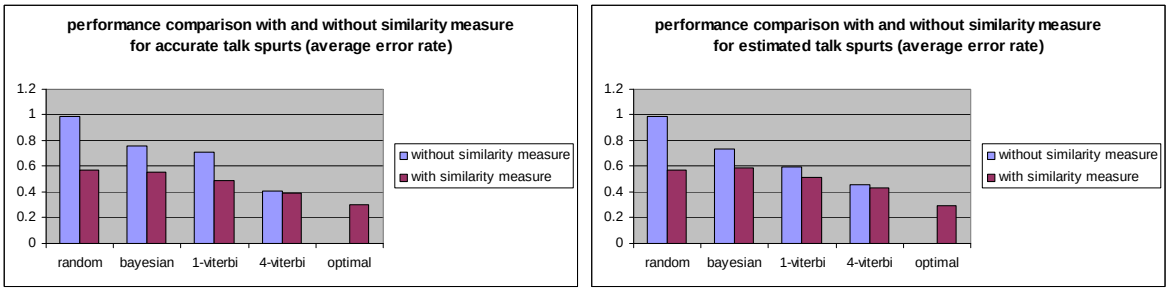

Fig. 19. Similarity measure improvements for symbols with accurate and estimated talk spurts of symbols from Rhyme 2 and Rhyme 3

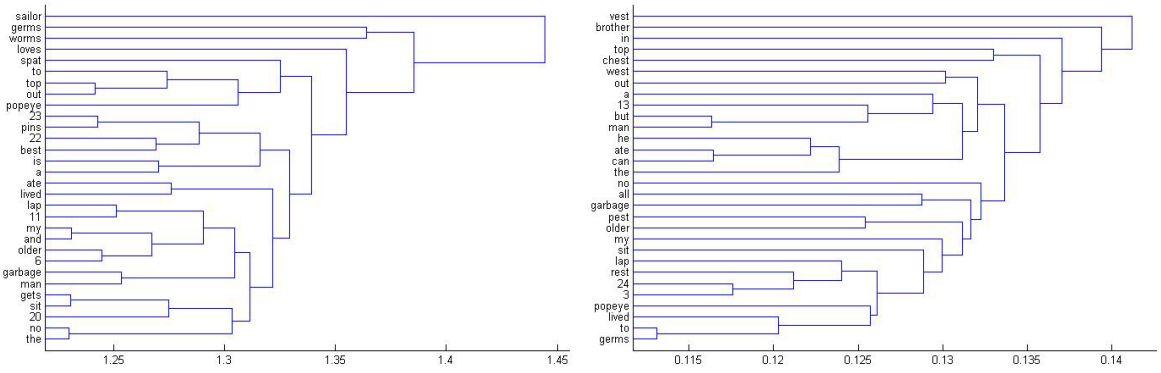

Fig. 20. Dendrograms for Bhattacharyya distances and error rates for symbols of Rhyme 2 and Rhyme 3

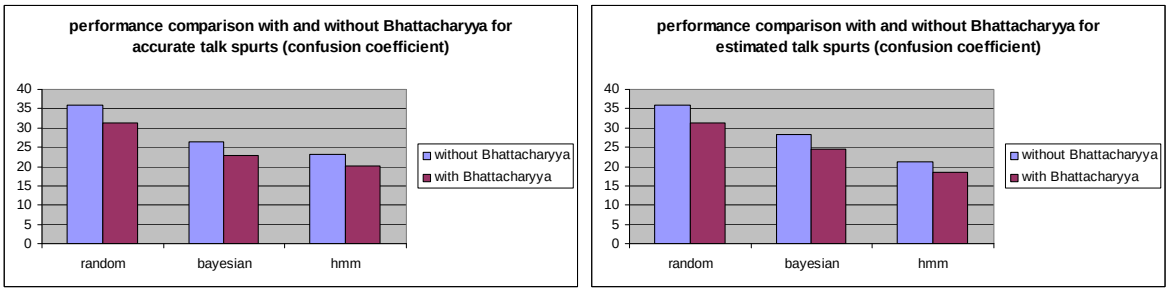

Fig. 21. Bhattacharyya improvements for symbols with accurate and estimated talk spurts of symbols from Rhyme 2 and Rhyme 3

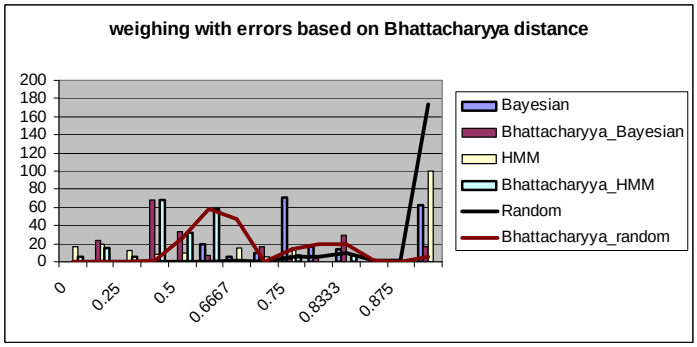

Fig. 22. Weighing the originally obtained errors with Bhattacharyya distance based errors

APPENDIX B

MATLAB CODE

The code structure used for HMM and Bayesian classification is provided here for reference.

```
%%%%%HMM and Bayesian classifier code in the same file%%%%
for num_tests=1:1
    warning off;
    test_talklengths=xlsread('testing_talkspurts.xls');
    %%%%%loading pre-calculated gaussian distributions
    %%%%%for the symbols of Rhyme 1
    load Rhyme1_accurate_mean_std_talklengths.mat
    %%%test sequences formed from symbols of Rhyme 1
    testinstances=xlsread('testcases.xls');
    %%%%%%corresponding talk spurt lengths of the
    %%%%%test sequences above
    %number of symbols
    num_words=size(meanWords,2);
    num_testinstances=num_tests;
    num_testcases=size(testinstances,1);
    for num_loops=1:200
        %randomly picking instances for testing
        randorder=randperm(num_testcases);
        var=length(randorder);
        training=testinstances(randorder
```

```
(1:var-num_testinstances),:);
testsequence=testinstances(randorder
(var-num_testinstances+1:var),:);
testing=test_talklengths(randorder
(var-num_testinstances+1:var),:);
%%%transition probabilities from training sentences
paircounts(1:num_words,1:num_words)=0;
x=training;
j=1;
for i=1:size(training,1)
    while(j<=size(training,2))
        if(j==size(training,2))
            break;
        end
        if((x(i,j)=0)&(x(i,j+1)=0))
            paircounts(x(i,j),x(i,j+1))
            =paircounts(x(i,j),x(i,j+1))+1;
        end
        j=j+1;
    end
    j=1;
end
prob_transition=[];
for i=1:num_words
    for j=1:num_words
        prob_transition(i,j)
```

```
        =paircounts(i,j)/sum(paircounts(i,:));
    end
end
%calculating word counts
wordcounts(1:num_words,1)=0;
x=training;
for i=1:size(training,1)
    for j=1:size(training,2)
        if(x(i,j)=0)
            wordcounts(x(i,j))=wordcounts(x(i,j))+1;
        end
    end
end
%prior probabilities of states(words)
for i=1:num_words
    prob_states(i,1)=wordcounts(i,1)/sum(wordcounts);
end
%%%%%HMM and Bayesian classification
for k=1:num_testinstances
    testData=testing(k,find(testing(k,:)=0));
    testseq=testsequence(k,find(testsequence(k,:)=0));
    sequence_length=length(testData);
    for i=1:sequence_length
        for j=1:num_words
            B(i,j)=(1/(stdWords(1,j)*sqrt(2*pi)))*
            exp(-((testData(i)-meanWords(1,j))^2)/
```

```
        (2*stdWords(1,j)*stdWords(1,j)));
    end
end
if (num_tests==1)
    posteriors=[];
    result=[];
    for len=1:sequence_length
        for nw=1:num_words
            posteriors(nw)=B(len,nw)*prob_states(nw);
        end
        [value index]=max(posteriors);
        result=[result index]
    end
    bayesian_errors(num_loops)=
    length(find(result=testseq))/length(path);
end
vit4_paths=
nviterbi_path(prob_states',prob_transition,B');
for z=1:size(vit4_paths,1)
    temp_vit4_err(z)=
    length(find(vit4_paths(z,:)=testseq))
    /length(path);
end
vit4_err(num_loops,k)=min(temp_vit4_err);
clear B;
vit1_errors(num_loops,k)=temp_vit4_err(1);
```

```
        end
    end
end
```

Dominant feature vectors based Similarity measure is calculated using the following code.

```
function sim=similarity_measure(signal1,signal2,fs)

%% number of eigens considered
num_eigens=8;
winlen=30/1000*fs %%%%%30ms long window
winoverlap=ceil(winlen/2);%%50% over lap

%% computing features
mfcc1=melfcc2(signal1,fs,'wintime', 0.03, 'hoptime',0.015);
ste1=short_time_energy(signal1,winlen,winoverlap);
zcr1=zero_cross_rate(signal1,winlen,winoverlap);
mfcc2=melfcc2(signal2,fs,'wintime', 0.03, 'hoptime',0.015);
ste2=short_time_energy(signal2,winlen,winoverlap);
zcr2=zero_cross_rate(signal2,winlen,winoverlap);

features1=[mfcc1;ste1;zcr1];
features2=[mfcc2;ste2;zcr2];

%%%%normalizing the features
features=[features1,features2];
```

```
[norm_features,xmean,xstd]=myMapStd(features');
temp1=myMapStd(features1',xmean,xstd);
features1=temp1';
temp2=myMapStd(features2',xmean,xstd);
features2=temp2';

%%%% eigen decomposition
covariance1=cov(features1');
[eigen_vectors1,eigen_values1]=eigs(covariance1,num_eigens);
eigen_values1 = diag(eigen_values1);

covariance2=cov(features2');
[eigen_vectors2,eigen_values2]=eigs(covariance2,num_eigens);
eigen_values2 = diag(eigen_values2);

%% weights
weights1=zeros(num_eigens,1); weights2=zeros(num_eigens,1);
for i=1:num_eigens
    weights1(i,1)=eigen_values1(i,1)/sqrt(sum(eigen_values1.*eigen_values1));
    weights2(i,1)=eigen_values2(i,1)/sqrt(sum(eigen_values2.*eigen_values2));
end

weights=weights1*weights2';

%% similarity between individual features
s=zeros(num_eigens,num_eigens);
```

```
for i=1:num_eigens
    for j=1:num_eigens
        temp=eigen_vectors1(:,i)'*eigen_vectors2(:,j);
        s(i,j)=(temp)^2;
    end
end

%%%similarity measure
sim=0; for i=1:num_eigens
    for j=1:num_eigens
        sim=sim+s(i,j)*weights(i,j);
    end
end
```

VITA

Tuneesh Kumar Lella received his Bachelor of Engineering in Computer Science and Master of Science in Mathematics from Birla Institute of Technology and Science, Pilani, India in June 2006. He entered the Computer Science program at Texas A & M University in August 2006, and since then, he has been pursuing his Master of Science degree receiving it in August 2008. He will begin work as a software engineer for Citrix Systems in Santa Clara, CA in August 2008. His e-mail address is saituneesh@tamu.edu.

The typist for this thesis was Tuneesh Kumar Lella.